Sparse Bayesian Learning and the Relevance Multi-Layer Perceptron Network

Gavin C. Cawley
School of Computing Sciences
University of East Anglia
Norwich NR4 7TJ U.K.
E-mail: gcc@cmp.uea.ac.uk

Nicola L. C. Talbot
School of Computing Sciences
University of East Anglia
Norwich NR4 7TJ U.K.
E-mail: nlct@cmp.uea.ac.uk

***Abstract*— We introduce a simple framework for sparse Bayesian learning with Multi-Layer Perceptron (MLP) networks, inspired by Tipping's Relevance Vector Machine (RVM). Like the RVM, a Bayesian prior is adopted that includes separate hyper-parameters for each weight, allowing redundant weights and hidden layer units to be identified and subsequently pruned from the network, whilst also providing a means to avoid over-fitting the training data. This approach is also more easily implemented, as only the diagonal elements of the Hessian matrix are used in the update formula for the regularisation parameters, rather than the traces of square sub-matrices of the Hessian corresponding to the weights associated with each regularisation parameter. The proposed Relevance Multi-Layer Perceptron (RMLP) is evaluated over several publicly available benchmark datasets, demonstrating the viability of the approach, giving rise to similar generalisation performance, but with far fewer weights.**

I. Introduction

Over-fitting of the training data is perhaps the most important issue to be addressed in applying flexible non-linear models, such as the multi-layer perceptron network. Structural stabilisation and formal regularisation are the two most commonly adopted approaches taken to avoid over-fitting in modern applications of artificial neural networks. Structural stabilisation aims to control the complexity of a neural network model by limiting the number of weights and/or hidden layer neurons comprising the model. Structural stabilisation is often implemented via constructive methods; starting with a small network, weights and/or hidden layer neurons are incrementally added to the network until some estimate of generalisation performance ceases to improve. Alternatively structural stabilisation can be achieved through pruning redundant weights and/or hidden units from an initially large network, according to some measure of the saliency of each weight in minimising the training criterion. Formal regularisation [1], on the other hand, seeks to control the complexity of the model by adding a term to the training criterion expected to penalise more complex models,

$$L = E_{\mathcal{D}} + \alpha E_{\mathcal{W}}, \tag{1}$$

where α is a regularisation parameter, controlling the relative importance of the original training criterion, $E_{\mathcal{D}}$, and the regularisation term, $E_{\mathcal{W}}$. Matching the complexity of the model to the difficulty of the learning task [2], then becomes a matter of finding the optimal value for the scalar regularisation parameter α. In practise, it is common to assign each weights into one of a set of C disjoint regularisation classes, each with controlled by its own regularisation parameter,

$$L = E_{\mathcal{D}} + \sum_{i=1}^{C} \alpha_i E_{\mathcal{W}}^i. \tag{2}$$

This allows, for instance, individual control of the regularisation of weights in different layers of a multi-layer perceptron network.

The regularised training criteria (1) and (2) admit a Bayesian interpretation, where $E_{\mathcal{D}}$ represents the negative logarithm of the likelihood of the data, given the model, and $E_{\mathcal{W}}$ represents the negative logarithm of a prior over the model parameters [3, 4]. The *evidence* framework [5–7] is a practical Bayesian training algorithm for multi-layer perceptron networks. The evidence framework provides a means of estimating appropriate values for the regularisation parameters, using only the training data, through maximisation of their marginal likelihood. In this paper we apply the evidence framework to multi-layer perceptron networks where the weight-decay regularisation term contains a separate regularisation parameter for each weight, e.g.

$$L = E_{\mathcal{D}} + \sum_{i=1}^{W} \alpha_i w_i^2. \tag{3}$$

The benefit of this approach is that the regularisation parameter for any weight that does not usefully contribute to reducing the data misfit becomes large, forcing the value of the corresponding weight to zero, enabling it to be pruned from the network. Over-fitting is then avoided via a combination of both formal regularisation *and* structural stabilisation. We name this method the Relevance Multi-Layer Perceptron (RMLP) network in deference to the Relevance Vector Machine (RVM) [8], which could be considered as applying a similar approach to the Radial Basis Function (RBF) network.

II. Method

For ease of exposition, this paper is concerned only with the solution of two-class pattern recognition problems using multi-layer perceptron networks [9], with two layers of modifiable weights, although the methods introduced are applicable to

arbitrarily complex feed-forward network structures and can also be applied in the context of regression. For two-class pattern recognition, with a logistic activation function in the output neuron, the output of the network is given by,

$$f(\boldsymbol{x};\boldsymbol{w}) = \frac{1}{1-\exp\{z(\boldsymbol{x};\boldsymbol{w})\}},$$

where

$$z(\boldsymbol{x};\boldsymbol{w}) = \sum_{i=1}^{M} \hat{w}_i \tanh\left(\sum_{j=1}^{d} w_{ij} + w_{i0}\right) + \hat{w}_0.$$

Given labelled training data, $\mathcal{D} = \{(\boldsymbol{x}_i, t_i)\}_{i=1}^{N}$, $\boldsymbol{x}_i \in \mathbb{R}^d$, $t_i \in (0,\ 1)$, the weights of the neural network are determined by minimising a regularised [1] loss function,

$$L(\boldsymbol{w};\alpha) = E_{\mathcal{D}} + \alpha E_{\mathcal{W}}, \tag{4}$$

where $E_{\mathcal{D}}$ measures the data misfit, $E_{\mathcal{W}}$ is the regularisation term, and α is a regularisation parameter governing the bias-variance trade-off [2]. In the case of two-class pattern recognition, assuming the target data are drawn independently from a Bernoulli distribution conditioned on the input data, it is appropriate to measure the data misfit using the cross-entropy error metric,

$$E_{\mathcal{D}} = -\sum_{i=1}^{\ell} \{t_i \log y_i + (1-t_i)\log(1-y_i)\}, \tag{5}$$

where $y_i = f(\boldsymbol{x}_i;\boldsymbol{w})$. The output of the network can then be interpreted as a consistent estimate of the *a-posteriori* probability of class membership, i.e. $f(\boldsymbol{x}) \approx p(t=1|\boldsymbol{x})$. The most common form of regularisation is known as "weight decay", where

$$E_{\mathcal{W}} = \sum_{i=1}^{N} w_i^2. \tag{6}$$

In the remainder of this section, we provide an overview of Bayesian training of multilayer perceptron networks under the evidence framework [5–7], before discussing the advantages and disadvantages of different regularisation terms.

A. Bayesian Learning within the Evidence Framework

In this section, we briefly summarise the Bayesian methods introduced by MacKay [5–7], based on the lucid exposition provided by Bishop [9]. Minimising the criterion given in equation (4) is equivalent to maximising the posterior distribution

$$p(\boldsymbol{w}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{w})p(\boldsymbol{w}|\alpha)}{p(\mathcal{D})}, \tag{7}$$

where the likelihood is given by the Bernoulli distribution,

$$p(\mathcal{D}|\boldsymbol{w}) = \prod_{i=1}^{\ell} y_i^{t_i}[1-y_i]^{(1-t_i)},$$

and the prior over model parameters by a multivariate Gaussian distribution,

$$p(\boldsymbol{w};\alpha) = \left[\frac{\alpha}{2\pi}\right]^{W/2} \exp\left\{-\frac{\alpha}{2}\|\boldsymbol{w}\|^2\right\},$$

where W is the number of weights. The Taylor expansion of $L(\boldsymbol{w},\alpha)$ around the most probable value, $\boldsymbol{w}^{\mathrm{MP}}$, gives rise to familiar Gaussian approximation to the posterior distribution, known as the "Laplace approximation",

$$p(\boldsymbol{w}|\mathcal{D}) \approx \frac{1}{Z^*}\exp\left\{-L\left(\boldsymbol{w}^{\mathrm{MP}}\right) - \frac{1}{2}\Delta\boldsymbol{w}^T\boldsymbol{A}\Delta\boldsymbol{w}\right\}, \tag{8}$$

where Z^* is an appropriate normalising constant, $\Delta\boldsymbol{w} = \boldsymbol{w} - \boldsymbol{w}^{\mathrm{MP}}$ and $\boldsymbol{A} = \nabla\nabla L(\boldsymbol{w};\alpha) = \nabla\nabla E_{\mathcal{D}} + \alpha\boldsymbol{I}$ is the Hessian of $L(\boldsymbol{w};\alpha)$ with respect to $\boldsymbol{w}$, evaluated at $\boldsymbol{w}^{\mathrm{MP}}$. The posterior distribution over the model parameters describes the uncertainty in estimating the model parameters from a finite set of training patterns. The Bayesian approach seeks to integrate out the model parameters when making inferences in order to account for the uncertainty in estimating the model parameters, such that

$$p(t=1|\boldsymbol{x},\mathcal{D}) = \int p(t=1|\boldsymbol{x},\boldsymbol{w})p(\boldsymbol{w}|\mathcal{D})dw.$$

This process is known as *marginalisation*. As $z(\boldsymbol{x};\boldsymbol{w})$ is a linear function of the model parameters, $\boldsymbol{w}$, the Laplace approximation implies that $z(\boldsymbol{x};\boldsymbol{w})$ also has a Gaussian distribution, centred on the most probable value, z_{MP},

$$p(z|\boldsymbol{x},\mathcal{D}) = \frac{1}{\sqrt{2\pi}s}\exp\left\{-\frac{(z-z_{\mathrm{MP}})^2}{2s^2}\right\},$$

with variance $s^2 = \boldsymbol{g}^T\boldsymbol{A}^{-1}\boldsymbol{g}$, where $\boldsymbol{g}$ is the first derivative of z, with respect to $\boldsymbol{w}$, evaluated at $\boldsymbol{w}^{\mathrm{MP}}$. Rather than marginalise over $\boldsymbol{\beta}$, we may equivalently marginalise over a, the probability that a pattern, $\boldsymbol{x}$, belongs to class for which $t=1$ can then be written as

$$\begin{aligned} p(t=1|\boldsymbol{x},\mathcal{D}) &= \int p(\mathcal{C}_1|z)p(z|\boldsymbol{x},\mathcal{D})dz \\ &= \int g(z)p(z|\boldsymbol{x},\mathcal{D})dz, \end{aligned} \tag{9}$$

where $g(z) = 1/[1+\exp(-z)]$. The integral (9) is not analytically tractable, and so MacKay [7] suggests the following approximation,

$$p(t=1;\boldsymbol{x},\mathcal{D}) \approx g(\kappa(s)z_{\mathrm{MP}}),$$

where

$$\kappa(s) = \left(1+\frac{\pi s^2}{8}\right)^{-\frac{1}{2}}.$$

The process of marginalisation can alternatively be implemented more accurately via Markov Chain Monte Carlo (MCMC) methods [10, 11].

The evidence approximation of [5–7] assumes that the posterior distribution for the regularisation parameter, $p(\alpha|\mathcal{D})$, is sharply peaked about its most probable value, α^{MP}, suggesting the following approximation to the posterior distribution for $\boldsymbol{w}$,

$$p(\boldsymbol{w}|\mathcal{D}) = \int p(\boldsymbol{w}|\alpha,\mathcal{D})p(\alpha|\mathcal{D})d\alpha \approx p(\boldsymbol{w}|\alpha^{\mathrm{MP}},\mathcal{D}).$$

Thus, rather than integrate out the regularisation parameter entirely (e.g. Buntine and Weigend [3, 4]), we simply proceed

with the analysis using the regularisation parameter fixed at its most likely value. For a discussion of the validity of this approach, see MacKay [12]. We seek therefore to maximise the posterior distribution,

$$p(\alpha|\mathcal{D}) = \frac{p(\mathcal{D}|\alpha)p(\alpha)}{p(\mathcal{D})}.$$

If the prior, $p(\alpha)$ is relatively insensitive to the value α, then maximising the posterior is approximately equivalent to maximising the likelihood term, $p(\mathcal{D}|\alpha)$, known as the *evidence* for α. Adopting the Gaussian approximation to the posterior for the model parameters, the log-evidence is given by

$$\log p(\mathcal{D}|\alpha) = -E_{\mathcal{D}}^{\mathrm{MP}} - \alpha E_{\mathcal{W}}^{\mathrm{MP}} - \frac{1}{2}\log|\boldsymbol{A}| + \frac{W}{2}\log\alpha. \quad (10)$$

Noting that $\boldsymbol{A} = \boldsymbol{H} + \alpha\boldsymbol{I}$, where $\boldsymbol{H}$ is the Hessian of $E_{\mathcal{D}}$ with respect to $\boldsymbol{w}$, if the eigenvalues of $\boldsymbol{H}$ are $\lambda_1, \lambda_2, \ldots, \lambda_W$, then the eigenvalues of $\boldsymbol{A}$ are $(\lambda_1 + \alpha), (\lambda_2 + \alpha), \ldots, (\lambda_W + \alpha)$. The derivative of $\log|\boldsymbol{A}|$ with respect to α (assuming that the eigenvalues of $\boldsymbol{H}$ are independent of α) is then given by

$$\frac{d}{d\alpha}\log|\boldsymbol{A}| = \frac{d}{d\alpha}\log\left\{\prod_{i=1}^{W}(\lambda_i + \alpha)\right\} = \sum_{i=1}^{W}\frac{1}{\lambda_i + \alpha}.$$

Setting the derivative of the log-evidence with respect to α to zero, we have

$$2\alpha E_{\mathcal{W}}^{\mathrm{MP}} = W - \sum_{i=1}^{W}\frac{\mu}{\lambda_i + \alpha} = \sum_{i=1}^{W}\frac{\lambda_i}{\lambda_i + \alpha} = \gamma,$$

where γ is the number of well determined parameters in the model. This leads to a simple update formula for the regularisation parameter:

$$\alpha^{\mathrm{new}} = \frac{\gamma}{2E_{\mathcal{W}}^{\mathrm{MP}}}. \quad (11)$$

The training procedure then alternates between updates of the primary model parameters $\boldsymbol{w}$ using, for instance, the method of scaled conjugate gradient descent, and updates of the regularisation parameter, α, according to equation (11).

B. Choice of Regularisation Term

Assuming the use of a simple weight-decay regularisation term, the simplest form for the regularised loss is given by,

$$L(\boldsymbol{w};\alpha) = E_{\mathcal{D}} + \alpha\sum_{i=1}^{W} w_i^2.$$

The regularisation parameter, α, essentially controls the average *scale* of the weights; if α is large, then the weights will on average be small, if α is small, then the weights are free to become large in the interests of minimising the data misfit $E_{\mathcal{D}}$. It is difficult to find a convincing argument why the average scale of the weights should be the same in different parts of the network. As a result it is common to assign groups of related weights into one of a number of *regularisation classes* (e.g. [4]), each governed by a separate regularisation parameter,

$$L(\boldsymbol{w};\boldsymbol{\alpha}) = E_{\mathcal{D}} + \sum_{i=1}^{C}\alpha_i\sum_{j=1}^{W_i}\left[w_j^i\right]^2.$$

where C is the number of regularisation classes and W_i is the number of weights in the i^{th} class. A common arrangement has one class comprising all input-to-hidden layer weights, and separate regularisation classes for the weights associated with each output unit. Again the vector of regularisation parameters can be updated under the evidence framework, except the number of well defined weights in each regularisation class is computed from the trace of a square sub-matrix of the inverse of the Hessian associated with those weights. A more complex arrangement, known as an Automatic Relevance Determination (ARD) prior [13] also places weights originating from each input unit into different regularisation classes. It has been observed that if an input feature does not significantly contribute to minimising the data misfit term, the evidence framework will set the regularisation parameter for the corresponding regularisation class to a very large value. This in turn will lead to the weights from the redundant input feature being forced to values close zero, and the corresponding input unit can be pruned from the network.

The Relevance Vector Machine (RVM) [8] can be viewed as a Radial Basis Function (RBF) network, with a basis function centred on each training pattern, minimising a regularised loss function imposing an ARD prior over the weights. As a result, the RVM is able to form a parsimonious model, comprised of basis functions centred on only a small fraction of the training patterns. In this paper, taking our inspiration from the relevance vector machine, we propose a regularisation term where each weight is assigned its own regularisation class,

$$L(\boldsymbol{w};\boldsymbol{\alpha}) = E_{\mathcal{D}} + \sum_{i=1}^{W}\alpha_i w_i^2,$$

and so there is one regularisation parameter for each weight. This arrangement has the advantage that individual redundant weights can easily be identified and pruned from the network (and by extension redundant hidden layer units and input features). This approach is also particularly easy to implement, if the diagonal approximation to the Hessian is adopted, only the diagonal elements need by evaluated in order to update the regularisation parameters, and the computation of the trace also becomes trivial (as the sub-matrices involved are now scalars). The disadvantage of this approach is that we must now estimate the number of very many more regularisation parameters, and so may need many more iterations until convergence. We refer to this arrangement as a Relevance Multi-Layer Perceptron (RMLP) in deference to the RVM.

III. Results

The relevance and conventional multi-layer perceptron networks were evaluated over a suite of seven well known real-world and synthetic benchmark datasets [14, 15]. The conventional and relevance multi-layer perceptron networks were

implemented in the MATLAB programming environment, using the NETLAB[1] toolbox [16]. As the training criteria for multi-layer perceptron networks typically exhibit many sub-optimal local minima, twenty networks were trained on each benchmark with random initialisation of the weights. The networks contained a single hidden layer, initially comprised of sixteen units. For all datasets, the regularisation parameters were updated at most 30 times, the weights being optimised via at most 100 iterations of scaled conjugate gradient descent between each update. The weights of the conventional Bayesian multi-layer perceptron (BMLP) networks were arranged into two regularisation classes, the first comprised of the input-to-hidden layer weights, and the second comprised of the weights feeding the lone output layer unit. The weights of the relevance multi-layer perceptron were pruned from the network if the value of γ for that weight fell below 1^{-6}.

TABLE I

MEAN ERROR RATE FOR BAYESIAN (BMLP) AND RELEVANCE (RMLP) MULTI-LAYER PERCEPTRON NETWORKS OVER 20 REPLICATES.

Benchmark	BMLP	RMLP
Breast cancer	$28.96\% \pm 0.17$	$27.47\% \pm 0.34$
Diabetis	$24.83\% \pm 0.15$	$24.61\% \pm 0.16$
Heart	$19.00\% \pm 0.00$	$19.15\% \pm 0.11$
Pima	$20.54\% \pm 0.04$	$20.53\% \pm 0.06$
Synthetic	$9.71\% \pm 0.01$	$9.44\% \pm 0.02$
Thyroid	$2.67\% \pm 0.00$	$2.67\% \pm 0.00$
Titanic	$21.84\% \pm 0.00$	$22.57\% \pm 0.00$

Table I shows the mean test set error rate over the 20 replicates for conventional Bayesian (BMLP) and relevance (RMLP) multi-layer perceptron networks, along with associates standard error of the mean. The performance of the BMLP and RMLP are generally similar, with the BMLP on average performing best on two datasets and the RMLP best on four with one tie. The differences in performance are however generally quite small and unlikely to be statistically significant.

TABLE II

MEAN NUMBER OF WELL-DEFINED WEIGHTS FOR BAYESIAN (BMLP) AND RELEVANCE (RMLP) MULTI-LAYER PERCEPTRON NETWORKS OVER 20 REPLICATES.

Benchmark	BMLP	RMLP
Breast cancer	33.96 ± 7.19	7.83 ± 0.58
Diabetis	71.24 ± 0.72	37.31 ± 1.21
Heart	27.02 ± 0.06	7.43 ± 0.16
Pima	19.11 ± 0.04	8.06 ± 0.11
Synthetic	43.71 ± 1.94	5.33 ± 0.11
Thyroid	21.71 ± 0.00	9.12 ± 0.11
Titanic	13.84 ± 0.00	3.83 ± 0.04

Table II shows the mean number of well defined weights of trained Bayesian and relevance multi-layer perceptron networks, and the associated standard error of the mean, over the twenty replicates performed for each benchmark. It is clear that the regularisation term involved in the relevance multi-layer perceptron is highly effective in identifying and pruning redundant weights, and the final models are generally very compact. Note this is achieved without a significant sacrifice in generalisation performance.

TABLE III

MEAN TRAINING TIME FOR BAYESIAN (BMLP) AND RELEVANCE (RMLP) MULTI-LAYER PERCEPTRON NETWORKS OVER 20 REPLICATES.

Benchmark	BMLP	RMLP
Breast cancer	$65.44s \pm 1.10$	$93.30s \pm 1.30$
Diabetis	$102.93s \pm 1.43$	$130.13s \pm 1.23$
Heart	$84.21s \pm 1.56$	$100.93s \pm 1.37$
Pima	$47.95s \pm 0.66$	$66.32s \pm 1.42$
Synthetic	$31.81s \pm 0.99$	$38.04s \pm 0.80$
Thyroid	$28.65s \pm 0.34$	$42.36s \pm 0.56$
Titanic	$12.37s \pm 0.19$	$21.80s \pm 0.72$

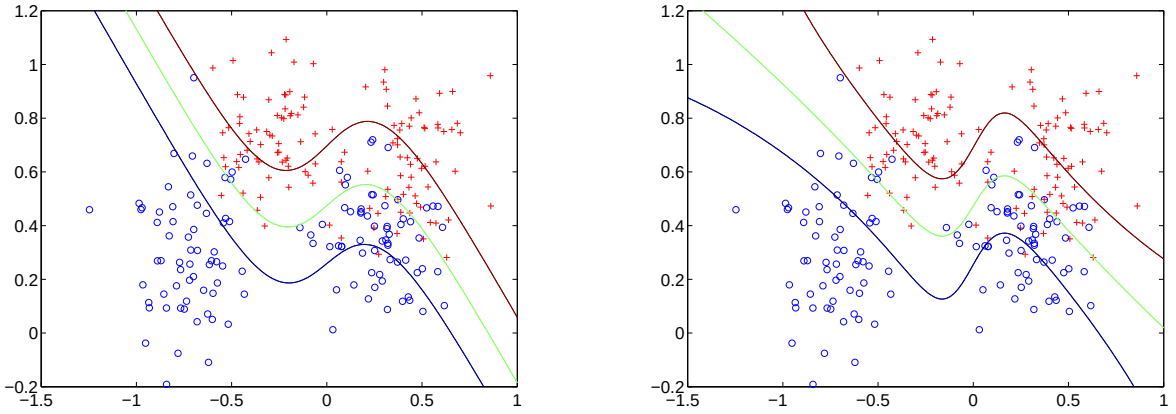
Table III shows the mean training time (and associated standard error of the mean) for the conventional Bayesian and relevance multi-layer perceptron networks. The total training time for the RMLP model is up to twice that of the BMLP. To some extent the computational expense of optimising one regularisation parameter for each weights is ameliorated by the reduction in the size of the network during training due to the pruning of redundant weights. It should be noted however, that the implementation of the RMLP has not been fully optimised, so for instance the diagonal elements of the Hessian are simply extracted from the full Hessian (computed using the NETLAB routine `nethess`), and could be computed more efficiently.

Figure 1 shows examples of B-MLP and R-MLP models of Ripley's `synthetic` benchmark dataset [14]. Both networks provide reasonable models of the data, however the R-MLP model is very much smaller, having on average only 5.33 well defined weights (the remainder having largely been pruned away) to the 43.71 of the B-MLP, while performing on average slightly better in terms of test set error (Table I).

IV. CONCLUSIONS

In this paper, we have proposed the use of weight-decay regularisation of multi-layer perceptron networks, with separate regularisation parameters for each weight, in the spirit of Tipping's Relevance Vector Machine (RVM). The resulting models are, like the RVM, highly sparse, without reducing generalisation performance or incurring unreasonably extended training time. We term the resulting model the Relevance Multi-Layer Perceptron (R-MLP) network. We hope that this prior will prove useful in practical applications of artificial neural networks, as it provides a means of simultaneously controlling over-fitting and generating a parsimonious model, two key aims of non-linear statistical modelling. We plan to apply these models to problems in the environmental sciences, such as statistical downscaling [17], involving the use of non-standard loss functions incorporating prior knowledge regarding the target distribution. Further work is also required

[1]Available from `http://www.ncrg.aston.ac.uk/netlab/`. Additional functions implementing the RMLP will be made available from the author's web site.



(a) B-MLP

(b) R-MLP

Fig. 1. Example of conventional Bayesian (B-MLP) and relevance (R-MLP) models of Ripley's `synthetic` benchmark dataset [14].

to evaluate the R-MLP against other pruning algorithms [4, 18].

ACKNOWLEDGEMENTS

This work was supported by the European Commission under the fifth framework thematic programme on Energy, Environment and Sustainable Development as part of the STARDEX project (STAtistical and Regional Dynamical Downscaling of EXtremes for European regions, contract number EVK2-CT-2001-00115, `http://www.cru.uea.ac.uk/projects/stardex`).

REFERENCES

[1] A. N. Tikhonov and V. Y. Arsenin, *Solutions of ill-posed problems*, John Wiley, New York, 1977.

[2] S. Geman, E. Bienenstock, and R. Doursat, "Neural networks and the bias/variance dilemma," *Neural Computation*, vol. 4, no. 1, pp. 1–58, 1992.

[3] W. L. Buntine and A. S. Weigend, "Bayesian back-propagation," *Complex Systems*, vol. 5, pp. 603–643, 1991.

[4] P. M. Williams, "Bayesian regularization and pruning using a Laplace prior," *Neural Computation*, vol. 7, no. 1, pp. 117–143, 1995.

[5] D. J. C. MacKay, "Bayesian interpolation," *Neural Computation*, vol. 4, no. 3, pp. 415–447, 1992.

[6] D. J. C. MacKay, "A practical Bayesian framework for backprop networks," *Neural Computation*, vol. 4, no. 3, pp. 448–472, 1992.

[7] D. J. C. MacKay, "The evidence framework applied to classification networks," *Neural Computation*, vol. 4, no. 5, pp. 720–736, 1992.

[8] M. E. Tipping, "Sparse Bayesian learning and the Relevance Vector Machine," *Journal of Machine Learning Research*, vol. 1, pp. 211–214, 2001.

[9] C. M. Bishop, *Neural Networks for Pattern Recognition*, Clarendon Press, 1995.

[10] R. M. Neal, "Bayesian learning for neural networks," 1996.

[11] W. R. Gilks, S. Richardson, and D. J. Spiegelhalter, *Markov Chain Monte Carlo in practice*, Interdisciplinary Statistics. Chapman and Hall/CRC, 1996.

[12] D. J. C. MacKay, "Hyperparameters : optimise or integrate out?," in *Maximum Entropy and Bayesian Methods*, G. Heidbreder, Ed. Kluwer, 1994.

[13] D. J. C. MacKay, "Probable networks and plausible predictions - a review of practical Bayesian methods for supervised neural networks," *Network : Computation in Neural Systems*, vol. 6, pp. 469–505, 1995.

[14] B. D. Ripley, *Pattern recognition and neural networks*, Cambridge University Press, 1996.

[15] S. D. Bay, "The UCI KDD archive [`http://kdd.ics.uci.edu/`]," University of California, Department of Information and Computer Science, Irvine, CA, 1999.

[16] I. Nabney, *NETLAB : Algorithms for pattern recognition*, Springer-Verlag, 2001.

[17] G. C. Cawley, S. R. Dorling, P. D. Jones, and C. Goodess, "Statistical downscaling with artificial neural networks," in *Proceedings of the European Symposium on Artificial Neural Networks (ESANN-2003)*, Bruges, Belgium, Apr. 23–25 2003, pp. 167–172.

[18] R. Reed, "Pruning algorithms — a survey," *IEEE Transactions on Neural Networks*, vol. 4, no. 5, pp. 740–747, 1993.